

# SET: AN ALGORITHM FOR CONSISTENT MATRIX COMPLETION

Wei Dai and Olgica Milenkovic

Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign

Email: {weidai07,milenkovic}@illinois.edu

**Abstract**—A new algorithm, termed subspace evolution and transfer (SET), is proposed for solving the consistent matrix completion problem. In this setting, one is given a subset of the entries of a low-rank matrix, and asked to find *one* low-rank matrix consistent with the given observations. We show that this problem can be solved by searching for a column space that matches the observations. The corresponding algorithm consists of two parts — subspace evolution and subspace transfer. In the evolution part, we use a line search procedure to refine the column space. However, line search is not guaranteed to converge, as there may exist barriers along the search path that prevent the algorithm from reaching a global optimum. To address this problem, in the transfer part, we design mechanisms to detect barriers and transfer the estimated column space from one side of the barrier to the another. The SET algorithm exhibits excellent empirical performance for very low-rank matrices.

**Index Terms**—Matrix completion, subspace.

## I. INTRODUCTION

Suppose that we observe a subset of entries of a matrix. The matrix completion problem asks when and how the matrix can be *uniquely* recovered based on the observed entries. This reconstruction task is ill-posed and computationally intractable. However, if the data matrix is known to have low-rank, exact recovery can be accomplished in efficient manners, provided that sufficiently many entries are revealed. Low-rank matrix completion problem has received considerable interests due to its wide applications, see for example [5] for more details.

An efficient way to solve the completion problem is via convex relaxation. Instead of looking at rank-restricted matrices, one can search for the matrix with minimum nuclear norm, subject to data consistency constraints. Although in general nuclear norm minimization is not equivalent to rank minimization, the former approach recovers the same solution as the latter if the data matrix satisfies certain incoherence conditions [6]. More importantly, nuclear norm minimization can be accomplished by polynomial complexity algorithms, for example, semi-definite programming or singular value thresholding (SVT) [1].

There are other low-complexity alternatives. Based on the subspace pursuit (SP) and CoSaMP algorithms for compressive sensing [7], [8], the authors of [2] developed the so called ADMiRA algorithm. A modification of the power factorization

algorithm was used for matrix completion in [3]. Another approach for solving this problem, termed OptSpace, was described in [4].

The problem considered in this paper and its algorithmic solution differ from all previously published approaches. The problem at hand is to identify *one* low-rank matrix *consistent* with the observations. The solution may or may not be unique. In contrast, most results in matrix completion deal with the somewhat more restrictive requirement that the reconstruction is *unique*. Hence, our approach can be applied to scenarios where the matrix is highly under-sampled, and where potentially many consistent solutions exist. The relaxation on uniqueness allows for the empirically observed performance improvement over other completion techniques.

To solve the consistent matrix completion problem, we propose an algorithm, termed subspace evolution and transfer (SET). We show that the matrix completion problem can be solved by searching for a column (or row) space that matches the observations. As a result, optimization on the Grassmann manifold, i.e., subspace evolution, plays a central role in the algorithm. However, there may exist “barriers” along the search path that prevent subspace evolution from converging to a global optimum. To address this problem, in the subspace transfer part, we design mechanisms to detect and cross barriers. Empirical simulations demonstrate the excellent performance of the proposed algorithm.

Despite resembling the OptSpace algorithm [4] in terms of using optimization over Grassmann manifolds, our approach substantially differs from this algorithm. Searching over only one space (column or row space) represents one of the most significant differences: in OptSpace, one searches *both* column and row spaces simultaneously, which introduces numerical and analytical difficulties. Moreover, when optimizing over the column space, one has to take care of “barriers” that prevent the search procedure from converging to a global optimum, an issue that was not addressed before since it was obscured by simultaneous column and row space searches.

## II. CONSISTENT MATRIX COMPLETION

Let  $\mathbf{X} \in \mathbb{R}^{m \times n}$  be an unknown matrix with rank  $r \ll \min(m, n)$ , and let  $\Omega \subset [m] \times [n]$  be the set of indices of the observed entries, where  $[K] = \{1, 2, \dots, K\}$ . Define the projection operator  $\mathfrak{P}_\Omega : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$  by

$$\mathbf{X} \mapsto \mathbf{X}_\Omega, \text{ where } (\mathbf{X}_\Omega)_{i,j} = \begin{cases} \mathbf{X}_{i,j} & \text{if } (i, j) \in \Omega \\ 0 & \text{if } (i, j) \notin \Omega \end{cases}.$$

The authors would like to thank Dayu Huang for his help in designing the employed line-search procedure, and to acknowledge useful discussions with Yoram Bresler, Justin Haldar, Ely Kerman, Angelia Nedich, and Zoi Rapti. Furthermore, the authors would also like to thank the authors of [1]–[4] for providing online software packages for their matrix completion algorithms.

The *consistent matrix completion* problem is to find *one* rank- $r$  matrix  $\mathbf{X}'$  that is consistent with the observations  $\mathbf{X}_\Omega$ , i.e.,

$$(P0) : \text{find } \mathbf{X}' \text{ such that} \\ \text{rank}(\mathbf{X}') \leq r \text{ and } \mathfrak{P}_\Omega(\mathbf{X}') = \mathfrak{P}_\Omega(\mathbf{X}) = \mathbf{X}_\Omega. \quad (1)$$

This problem is well defined as  $\mathbf{X}_\Omega$  is generated from the matrix  $\mathbf{X}$  with rank  $r$  and therefore there must exist *at least one solution*. In this paper, like in other approaches in [2]–[4], we assume that the rank  $r$  is given. In practice, one may try to sequentially guess a rank bound until a satisfactory solution has been found.

### III. THE SET ALGORITHM

#### A. Why optimize over column spaces only?

In this section, we show that the problem (P0) is equivalent to finding a column space consistent with the observations.

Let  $\mathcal{U}_{m,r}$  be the set of  $m \times r$  matrices with  $r$  orthonormal columns, i.e.,  $\mathcal{U}_{m,r} = \{\mathbf{U} \in \mathbb{R}^{m \times r} : \mathbf{U}^T \mathbf{U} = \mathbf{I}_r\}$ . Define a function

$$f : \mathcal{U}_{m,r} \rightarrow \mathbb{R} \\ \mathbf{U} \mapsto \min_{\mathbf{W} \in \mathbb{R}^{n \times r}} \|\mathbf{X}_\Omega - \mathfrak{P}_\Omega(\mathbf{U}\mathbf{W}^T)\|_F^2, \quad (2)$$

where  $\|\cdot\|_F$  denotes the Frobenius norm. The function  $f$  captures the consistency between the matrix  $\mathbf{U}$  and the observations  $\mathbf{X}_\Omega$ : if  $f(\mathbf{U}) = 0$ , then there exists a matrix  $\mathbf{W}$  such that the rank- $r$  matrix  $\mathbf{U}\mathbf{W}^T$  satisfies  $\mathfrak{P}_\Omega(\mathbf{U}\mathbf{W}^T) = \mathbf{X}_\Omega$ . Hence, the consistent matrix completion problem is equivalent to

$$(P1) : \text{find } \mathbf{U} \in \mathcal{U}_{m,r} \text{ such that } f(\mathbf{U}) = 0. \quad (3)$$

The solution  $f(\mathbf{U})$  is not unique in the space  $\mathcal{U}_{m,r}$ . An important property of  $f$  is that  $f(\mathbf{U}) = f(\mathbf{U}\mathbf{V})$  for any  $r$ -by- $r$  orthogonal matrix  $\mathbf{V}$ , since  $\mathbf{U}\mathbf{W}^T = (\mathbf{U}\mathbf{V})(\mathbf{W}\mathbf{V}^T)^T$ . Hence, the function  $f$  depends only on the subspace spanned by the columns of  $\mathbf{U}$ , i.e., the  $\text{span}(\mathbf{U})$ . Note that all columns of the matrix of the form  $\mathbf{U}\mathbf{W}^T$  lie in the linear subspace  $\text{span}(\mathbf{U})$ . The consistent matrix completion problem is essentially *finding a column space consistent with the observed entries*.

We find the following definitions useful for the exposition to follow. The set of all  $r$ -dimensional linear subspaces in  $\mathbb{R}^n$  is called the Grassmann manifold, and is denoted by  $\mathcal{G}_{m,r}$ . Given a subspace  $\mathcal{U} \in \mathcal{G}_{m,r}$ , one can always find a matrix  $\mathbf{U} \in \mathcal{U}_{m,r}$ , such that  $\mathcal{U} = \text{span}(\mathbf{U})$ . The matrix  $\mathbf{U}$  is referred to as a generator matrix of  $\mathcal{U}$ . Although a given subspace  $\mathcal{U} \in \mathcal{G}_{m,r}$  has multiple generator matrices, a given matrix  $\mathbf{U} \in \mathcal{U}_{m,r}$  uniquely defines a subspace. For this reason, we henceforth use  $\mathbf{U}$  to represent its generated subspace.

#### B. The SET algorithm: a high level description

Our algorithm aims to minimize the objective function  $f(\mathbf{U})$ , provided that the minimum value of  $f(\mathbf{U})$  is *known to be zero*. Ideally, a solution can be obtained by using a line search procedure on the Grassmann manifold. Here, line search refers to iterative refinements of the interval in

which the function attains its minimum. Hence, the “subspace evolution” part of the algorithm reduces to a well studied optimization method.

The main difficulty that arises during line search, and makes the SET algorithm highly non-trivial is when during the search, one encounters “barriers”. Careful inspection reveals that the objective function  $f$  can be decomposed into a sum of atomic functions, each of which involves only one column of  $\mathbf{X}_\Omega$  (see Section III-D for details). Along the gradient descent path, these atomic functions may not agree with each other: some decrease and some increase. Increases of some atomic functions may result in “bumps” in the  $f$  curve, which block the search procedure from global optima and are therefore referred to as *barriers*. The main component of the “transfer” part of the algorithm is to identify whether there exist barriers along the gradient descent path. Detecting barriers is in general a very difficult task, since one does not know the locations of global minima. Nevertheless, we observe that barriers can be detected by the existence of atomic functions with inconsistent descent directions. When such a scenario is encountered, the algorithm “transfers” the starting point of line search to the other side of the barriers, and proceeds from there. Such a transfer does not overshoot global minima as we enforce consistency of the steep descent directions at the points before and after the transfer.

In summary, we start with a randomly generated  $\mathbf{U} \in \mathcal{U}_{m,r}$  and then refine it until  $f(\mathbf{U}) = 0$ . At each iteration, we first detect and then cross barriers if there are any, and then perform line search. The details of subspace evolution and transfer are given in Section III-C and III-D. Simulation results are presented in Section IV.

#### C. Subspace evolution

Due to space limitation, we focus on the  $r = 1$  case in Sections III-C and III-D. Furthermore, our exposition aims to make the algorithmic details as transparent to the readers as possible. The highly technical *performance and complexity analysis* of SET for *both*  $r = 1$  and  $r > 1$  is deferred to the journal version of the paper.

For the optimization problem at hand, we shall refine the current column space estimate  $\mathbf{u}$  following the gradient descent direction. Here, the lowercase letter  $\mathbf{u}$  is used to emphasize that the  $\mathbf{U}$  matrix is a vector when  $r = 1$ . Let  $\mathbf{w}_\mathbf{u}$  be a length- $n$  column vector that achieves  $f(\mathbf{u})$ , and let  $\mathbf{X}_r = \mathbf{X}_\Omega - \mathfrak{P}_\Omega(\mathbf{u}\mathbf{w}_\mathbf{u}^T)$ . Then the gradient<sup>1</sup> of  $f$  at  $\mathbf{u}$  is given by

$$\nabla_\mathbf{u} f = -2\mathbf{X}_r \mathbf{w}_\mathbf{u}. \quad (4)$$

The gradient descent path is chosen to be *the geodesic curve* on the Grassmann manifold with direction  $\mathbf{h} = -\nabla_\mathbf{u} f / \|\nabla_\mathbf{u} f\|_F$ . A geodesic curve is an analogue of a straight line in an Euclidean space: given two points on the manifold, the geodesic curve connecting them is the path of the shortest length on the manifold. According to [9, Theorem 2.3], the geodesic curve starting from  $\mathbf{u}$ , along  $\mathbf{h}$ , is given by

$$\mathbf{u}(t) = \mathbf{u} \cos t + \mathbf{h} \sin t, \quad t \in [0, \pi]. \quad (5)$$

<sup>1</sup>The gradient is well defined almost everywhere in  $\mathcal{U}_{m,r}$ .

We restrict  $t$  to the interval  $[0, \pi)$  because  $f(\mathbf{u}(t))$  has period  $\pi$ , i.e.,  $f(\mathbf{u}(t + \pi)) = f(-\mathbf{u}(t)) = f(\mathbf{u}(t))$ . Interested readers are referred to [9] for more details on geodesics on the Grassmann manifold.

The subspace evolution part is designed to search for a minimizer (in most cases, a local minimizer) of the function  $f$  along the geodesic curve. Our implementation includes two steps. The goal of the first step is to identify an interval  $[0, t_{\max}]$  that contains a minimizer. Since  $f(t)$  is periodic,  $t_{\max}$  is upper bounded by  $\pi$ . The second step is devoted to locating the minimizer  $t^* \in [0, t_{\max}]$  accurately by iteratively applying the golden section rule [10]. These two steps are described in Algorithm 1. The constants are set to  $\epsilon = 10^{-9}$ ,  $c_1 = (\sqrt{5} - 1)/2$ ,  $c_2 = c_1/(1 - c_1)$  and  $itN = 10$ . Ideally, the starting step size  $\epsilon > 0$  should be chosen as small as possible. We fix it to a constant as computers only have finite precision and  $10^{-9}$  is already sufficiently small in all our experiments.

---

**Algorithm 1** Subspace evolution.

---

**Input:**  $\mathbf{X}_\Omega$ ,  $\Omega$ ,  $\mathbf{u}$ , and  $itN$ .

**Output:**  $t^*$  and  $\mathbf{u}(t^*)$ .

**Step A:** find  $t_{\max} \leq \pi$  such that  $t^* \in [0, t_{\max}]$

Let  $t' = \epsilon\pi$ .

- 1) Let  $t'' = c_2 \cdot t'$ . If  $t'' > \pi$ , then  $t_{\max} = \pi$  and quit Step A.
- 2) If  $f(\mathbf{u}(t'')) > f(\mathbf{u}(t))$ , then  $t_{\max} = t''$  and quit Step A.
- 3) Otherwise,  $t' = t''$ . Go back to step 1).

**Step B:** numerically search for  $t^*$  in  $[0, t_{\max}]$ .

Let  $t_1 = t_{\max}/c_2^2$ ,  $t_2 = t_{\max}/c_2$ ,  $t_4 = t_{\max}$ , and  $t_3 = t_1 + c_1(t_4 - t_1)$ . Let  $itn = 1$ . Perform the following iterations.

- 1) If  $f(\mathbf{u}(t_1)) > f(\mathbf{u}(t_2)) > f(\mathbf{u}(t_3))$ , then  $t_1 = t_2$ ,  $t_2 = t_3$ , and  $t_3 = t_1 + c_1(t_4 - t_1)$ .
- 2) Else,  $t_4 = t_3$ ,  $t_3 = t_2$  and  $t_2 = t_1 + (1 - c_1)(t_4 - t_1)$ .
- 3)  $itn = itn + 1$ . If  $itn > itN$ , then quit the iterations. Otherwise, go back to step 1).

Let  $t^* = \arg \min_{t \in \{t_1, \dots, t_4\}} f(\mathbf{u}(t))$  and compute  $\mathbf{u}(t^*)$ .

---

#### D. Subspace transfer

Unfortunately, the objective function  $f(\mathbf{u})$  may not be a convex function of  $\mathbf{u}$ . The described linear search procedure may not converge to a global minimum because the search path may be blocked by what we call “barriers”. We show next how to overcome the problem introduced by barriers.

At this point, we formally introduce the decoupling principle: the objection function  $f(\mathbf{u}(t))$  is the squared Frobenius norm of the residue matrix; it can be decomposed as the sum of the squared Frobenius norm of the residue columns. More precisely, let  $\mathbf{x}_{\Omega_j} \in \mathbb{R}^{m \times 1}$  be the  $j^{\text{th}}$  column of the matrix  $\mathbf{X}_\Omega$ . Let  $\mathfrak{P}_{\Omega_j} : \mathbb{R}^{m \times 1} \rightarrow \mathbb{R}^{m \times 1}$  be the projection operator corresponding to the  $j^{\text{th}}$  column, defined by

$$\mathbf{v} \mapsto \mathbf{v}_{\Omega_j}, \text{ where } (\mathbf{v}_{\Omega_j})_i = \begin{cases} v_i & \text{if } (i, j) \in \Omega \\ 0 & \text{if } (i, j) \notin \Omega \end{cases}.$$

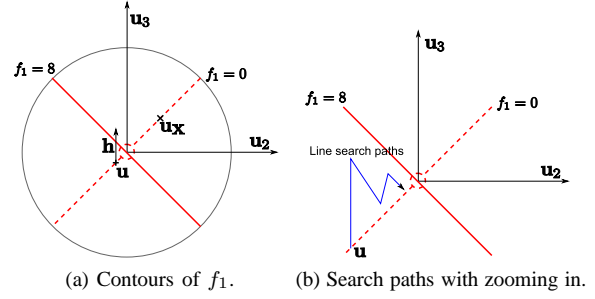


Figure 1: An illustrative example for barriers.

Then the objective function  $f(\mathbf{u}(t))$  can be written as a sum of  $n$  atomic functions:

$$\begin{aligned} f(\mathbf{u}(t)) &= \min_{\mathbf{w} \in \mathbb{R}^{n \times 1}} \|\mathbf{X}_\Omega - \mathfrak{P}_\Omega(\mathbf{u}\mathbf{w}^T)\|_F^2 \\ &= \sum_{j=1}^n \underbrace{\min_{\mathbf{w}_j \in \mathbb{R}} \|\mathbf{x}_{\Omega_j} - \mathfrak{P}_{\Omega_j}(\mathbf{u}(t)\mathbf{w}_j)\|_F^2}_{f_j(\mathbf{u}(t))}. \end{aligned} \quad (6)$$

This principle is essential to understanding the behavior of  $f(\mathbf{u}(t))$ .

The following example illustrates the concept of a barrier. Consider an incomplete observation of a rank-one matrix  $\begin{bmatrix} [?, 2, 2]^T, [2, ?, 1]^T \end{bmatrix}$ , where question marks denote that the corresponding entries are unknown. It is clear that the objective function  $f(\mathbf{u}(t))$  is minimized by  $\mathbf{u}_X = \frac{1}{\sqrt{6}} [2, 1, 1]^T$ , i.e.,  $f(\mathbf{u}_X) = 0$ . Suppose that one starts with the initial guess  $\mathbf{u} = \frac{1}{\sqrt{102}} [-10, 1, 1]^T$ . The contours of the atomic function  $f_1(\mathbf{u})$ , projected on the plane spanned by  $\mathbf{u}_2$  and  $\mathbf{u}_3$ , is depicted in Fig. 1a. All  $\mathbf{u}$ 's with  $\mathbf{u}_2 = -\mathbf{u}_3$  lie on the contour  $f_1(\mathbf{u}) = 8$ . Computations show that the gradient descent direction  $\mathbf{h}$  is pointing upward. However  $f(\mathbf{u}) = f_1(\mathbf{u}) + f_2(\mathbf{u}) = 0 + f_2(\mathbf{u}) \leq 5 < 8$ . Any gradient descent algorithms can not pass through the contour  $f_1(\mathbf{u}) = 8$ . Careful tracking of several line search steps (Fig. 1b) shows that  $\mathbf{u}(t)$  will approach  $[-1, 0, 0]^T$ , but will never cross the contour  $f_1 = 8$ . That is, the contour  $f_1 = 8$  forms a “barrier” for the line search procedure.

It is possible to detect barriers algorithmically. It can be verified that  $f_j(\mathbf{u}(t))$  has a unique minimizer and maximizer<sup>2</sup>, given by

$$t_{\max,j} = \arg \max_{t \in [0, \pi)} f_j(\mathbf{u}(t)) \text{ and } t_{\min,j} = \arg \min_{t \in [0, \pi)} f_j(\mathbf{u}(t)) \quad (7)$$

respectively. There are closed-form equations for these two quantities given an initial vector  $\mathbf{u}$  and a direction  $\mathbf{h}$ . We say that the  $k^{\text{th}}$  column of  $\mathbf{X}_\Omega$  forms a barrier if there exists a  $j \in [n]$ ,  $j \neq k$ , such that

- 1) the maximizer of  $f_k$  appears before the minimizer of  $f_j$ , i.e.,  $t_{\max,k} < t_{\min,j} < t_{\max,j}$ ; and
- 2) the gradients of  $f$  at  $\mathbf{u}(0)$  and  $\mathbf{u}(t_{\max,k})$  are consistent (form a sharp angle), i.e.,  $\frac{d}{dt} f(\mathbf{u}(t))|_{t=t_{\max,k}} < 0$ .

<sup>2</sup>The exception is that  $\mathfrak{P}_{\Omega_j}(\mathbf{u})$  and  $\mathfrak{P}_{\Omega_j}(\mathbf{h})$  are linearly dependent, which happens with zero probability and is ignored here for simplicity.

When a barrier is detected, we transfer  $\mathbf{u}$  from one side of it to the other. In our implementation, we focus on the closest barriers to  $\mathbf{u}$  to avoid overshooting. Define

$$\mathcal{J} = \{j : \text{the } j^{\text{th}} \text{ column of } \mathbf{X}_\Omega \text{ admits barriers}\},$$

$$j^* = \arg \min_{j \in \mathcal{J}} t_{p,j}, \text{ and} \quad (8)$$

$$k^* = \arg \max_k \left\{ t_{o,k} : \text{the } k^{\text{th}} \text{ column of } \mathbf{X}_\Omega \text{ forms a barrier for the } j^{*th} \text{ column of } \mathbf{X}_\Omega \right\}. \quad (9)$$

The subspace transfer part is described in Algorithm 2.

---

**Algorithm 2** Subspace transfer

---

**Input:**  $\mathbf{X}_\Omega$ ,  $\Omega$ , and  $\mathbf{u}$ .

**Output:**  $t_{st}$  and  $\mathbf{u}(t_{st})$ .

**Steps:**

- 1) Compute  $t_{o,j}$  and  $t_{p,j}$  for each column  $j$  satisfying  $\text{rank}([\mathbf{u}_{\Omega_j}, \mathbf{h}_{\Omega_j}]) = 2$ .
  - 2) Suppose that there exist barriers.
    - a) find  $j^*$  and  $k^*$  according to (8) and (9) respectively.
    - b) Let  $t_{st} = t_{o,k^*}$  and compute  $\mathbf{u}(t_{st})$ .
  - 3) Otherwise,  $t_{st} = 0$  and  $\mathbf{u}(t_{st}) = \mathbf{u}$ .
- 

#### IV. PERFORMANCE EVALUATION

Here, we introduce an error tolerance parameter  $\epsilon_e > 0$ . In practice, instead of requiring exact data matching, it usually suffices to have  $\|\mathfrak{P}_\Omega(\mathbf{X}') - \mathbf{X}_\Omega\|_F^2 < \epsilon_e \|\mathbf{X}_\Omega\|_F^2$  for some small  $\epsilon_e$ . In our simulations, we set  $\epsilon_e = 10^{-6}$ .

We tested the SET algorithm by randomly generating low-rank matrices  $\mathbf{X}$  and index sets  $\Omega$ . Specifically, we decompose the matrix  $\mathbf{X}$  into  $\mathbf{X} = \mathbf{U}_\mathbf{X} \mathbf{S}_\mathbf{X} \mathbf{V}_\mathbf{X}^T$ , where  $\mathbf{U}_\mathbf{X} \in \mathcal{U}_{m,r}$ ,  $\mathbf{V}_\mathbf{X} \in \mathcal{U}_{n,r}$ , and  $\mathbf{S}_\mathbf{X} \in \mathbb{R}^{r \times r}$ . We generate  $\mathbf{U}_\mathbf{X}$  and  $\mathbf{V}_\mathbf{X}$  from the isotropic distribution on the set  $\mathcal{U}_{m,r}$  and  $\mathcal{U}_{n,r}$ , respectively. The entries of the  $\mathbf{S}_\mathbf{X}$  matrix are independently drawn from the standard Gaussian distribution  $\mathcal{N}(0, 1)$ . This step is important in order to guarantee the randomness in the singular values of  $\mathbf{X}$ . The index set  $\Omega$  is randomly generated from the uniform distribution over the set  $\{\Omega' \subset [m] \times [n] : |\Omega'| = |\Omega|\}$ .

The performance of the SET algorithm is excellent. We tested different matrices with different ranks and different sampling rates, defined as  $|\Omega| / (m \times n)$ . The performance is shown in Fig. 2. The performance improvement due to the transfer step is significant. We also compare the SET algorithm to other matrix completion algorithms<sup>3</sup>. As shown in Figure 3, the SET algorithm outperforms all other tested completion approaches. For most realizations, the SET algorithm needs less than 500 iterations to converge. However, there are examples for which the reconstruction error is still large after 2000 iterations. Studying such realizations indicates that the

major reason for this phenomena is a slow convergence rate: after many iterations the barriers are still too far away from space  $\mathbf{U}$  to be detectable. One future research direction is therefore to speed up the SET algorithm. As a final remark, we notice that there exist a critical range of sampling rates, in which the performance deteriorates. As can be observed from the figures, this range shifts to the right as the rank increases.

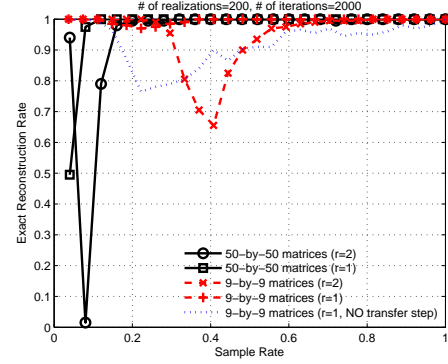


Figure 2: Performance of the SET algorithm.

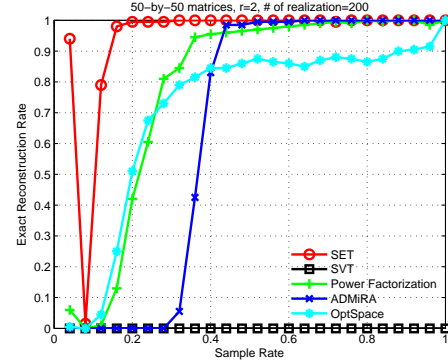


Figure 3: Performance comparison.

#### REFERENCES

- [1] J. Cai, E. J. Candes, and Z. Shen, "A singular value thresholding algorithm for matrix completion," Oct. 2008.
- [2] K. Lee and Y. Bresler, "ADMIRA: atomic decomposition for minimum rank approximation," Apr. 2009.
- [3] J. Haldar and D. Hernando, "Rank-constrained solutions to linear matrix equations using powerfactorization," *IEEE Signal Processing Letters*, pp. 16:584–587, 2009.
- [4] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," 2009.
- [5] E. Candes and B. Recht, "Exact matrix completion via convex optimization," *Submitted for publication*, 2008.
- [6] E. J. Candes and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," Mar. 2009.
- [7] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inform. Theory*, vol. 55, pp. 2230 – 2249, May 2009.
- [8] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, pp. 301–321, May 2009.
- [9] A. Edelman, T. Arias, S. T. Smith, Steven, and T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM Journal on Matrix Analysis and Applications*, vol. 20, pp. 303–353, April 1999.
- [10] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. Academic Press, 1982.

<sup>3</sup>Though the SVT algorithm is not designed to solve the problem (P0), we include it for completeness. In the standard SVT algorithm, there is no explicit constraint on the rank of the *reconstructed* matrix. For fair comparison, we take the best rank- $r$  approximation of the reconstructed matrix, and check whether it satisfies the performance criterion.